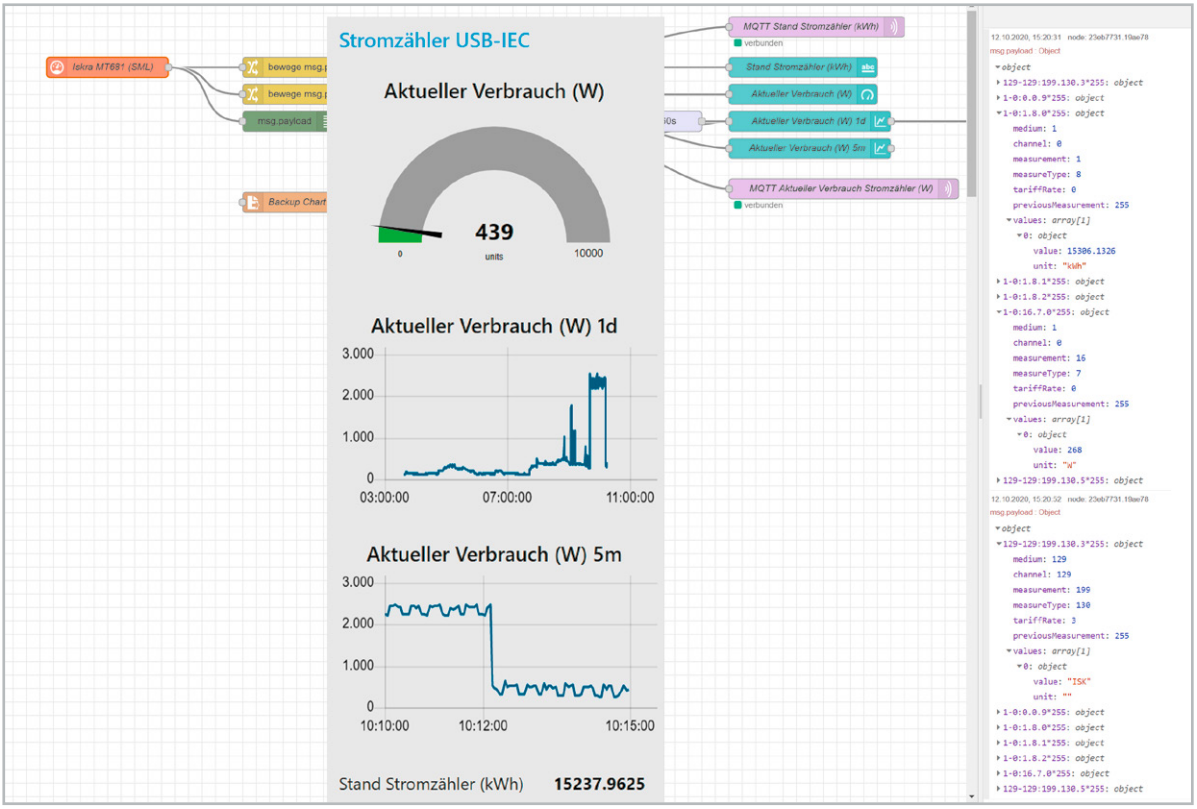
Intelligenter Strom

Daten vom USB-IEC-Interface mit Node-RED auslesen und verarbeiten

In dieser Ausgabe des ELVjournals stellen wir auf Seite 60 den neuen Bausatz USB-IEC-Interface zum Auslesen von Daten über optische Schnittstellen beispielsweise an Stromzählern vor. Um die Informationen zu erfassen und weiterzuverarbeiten, gibt es eine Reihe freier und kommerzieller Software. In diesem Beitrag zeigen wir, wie man mithilfe eines Raspberry Pi und des Software-Prototyping-Tools Node-RED die Daten auslesen, in einer Oberfläche optisch aufbereiten und anderen Anwendungen zur Verfügung stellen kann.



Grundlagen

Um Daten wie die aktuelle Leistung oder den Zählerstand über die optische Schnittstelle der neueren Stromzähler abzurufen, sollte man sich zunächst mit den Grundlagen dieser Technologie beschäftigen. Die in der Norm EN 62056-21 definierte Schnittstelle haben wir bereits im ELVjournal 2/2016 [1] ausführlich vorgestellt. Zudem haben wir das häufig verwendete SML-Protokoll in der Ausgabe 4/2019 [2] eingehend erläutert.

Als Grundlage für das Auslesen, die Verarbeitung und Anzeige sowie die Weiterleitung der Daten nutzen wir das vor allem für das Prototyping geeignete Software-Tool Node-RED. In unserer dreiteiligen Einführung im ELVjournal 4/2020 [3], 5/2020 [4] und dieser Ausgabe ab Seite 50 sind wir ausführlich auf die Grundlagen, die Installation und Anwendung dieser praktischen Software eingegangen.

Schritt 1 Hardware-Test

Als Grundlage bei der Hardware dient uns der neue Raspberry Pi 4 Model B mit 2 GB RAM [5]. Es dürften allerdings auch Raspberry Pi der zweiten und dritten Generation ausreichen, um Node-RED problemlos zu betreiben.

Für die ersten Schritte benötigen wir die Software allerdings noch nicht, da wir zunächst testen, ob überhaupt Daten aus der optischen Schnittstelle ausgegeben werden. Dazu bringen wir als Erstes das USB-IEC-Interface am Stromzähler an und schließen dann den USB-Stecker an den betriebsbereiten Raspberry Pi an. Ist die optische Schnittstelle aktiv, sollte die RX-LED des USB-IEC-Interface periodisch aufleuchten und damit den Empfang der Daten signalisieren (Bild 1).

Sind in dem Bausatz die LEDs nicht verbaut, kann man sich alternativ auch mit einer Smartphone-Kamera den Bereich anschauen, in dem sich die Übertragungs-LEDs des Stromzählers befinden. Obwohl diese LEDs im Infrarotbereich übertragen und für das menschliche Auge nicht sichtbar sind, können die Handy-Kameras diesen Wellenlängenbereich des Lichts erfassen. Auf diese Weise ist ebenfalls eine Kontrolle möglich, ob die optische Schnittstelle aktiviert ist.

Nach dem Hardware-Test teilen sich die Wege zum Auslesen der Daten von der Schnittstelle des Stromzählers und die Verarbeitung per Node-RED. Ist die optische Schnittstelle aktiviert und kommen die Daten per SML-Protokoll, können wir mit Schritt 4 fortfahren. Ansonsten müssen wir uns zunächst um die Aktivierung der Datenausgabe in Schritt 2 kümmern.

Hinweis: Es kann auch sein, dass der Zähler die Daten im Push-Betrieb ausgibt, es sich aber nicht um das SML-, sondern um das D0-Protokoll handelt (Modus D).

Schritt 2 Aktivierung der optischen Schnittstelle

Abhängig vom Typ des Stromzählers, der im Zählerschrank vom Energieversorger installiert ist, lässt sich die optische Schnittstelle auf unterschiedliche Arten aktivieren. Eine Suche im Internet hilft in der Regel weiter und auch Webseiten wie beispielsweise volkszaehler.org, die sich mit der Erfassung von Zählerständen befassen, haben hier unter Umständen wertvolle Informationen. Angaben zum verwendeten Protokoll zahlreicher Zähler findet man auch in der Konfigurationsliste des ES-IEC [6].

Wir wollen das „Aufwecken" der Schnittstelle, das in der Regel vor jedem Auslesen erneut ausgelöst werden muss, exemplarisch an einem Stromzähler vom Typ Iskra MT171 zeigen, der seine Daten basierend auf dem D0-Protokoll im Modus C ausgibt. Dazu schließen wir zunächst das USB-IEC-Interface sowohl an den Raspberry Pi per USB als auch mit dem Lesekopf an den Stromzähler an.

Zur Kommunikation mit dem Zähler über die serielle Schnittstelle bieten sich unter Raspbian beispielsweise cutecom und unter Windows HTerm [7] an. Da das USB-IEC-Interface ohnehin schon am Raspberry Pi angeschlossen ist, verwenden wir das Linux-Tool. Unter Windows ist die Vorgehensweise analog – entscheidend sind hier zum einen die Einstellungen für die Schnittstelle und zum anderen die Befehle, die an den Zähler geschickt werden.

Zunächst installieren wir mit *sudo apt install cutecom* die Software auf dem Raspberry Pi. Mit *dmesg* können wir in den Log-Einträgen die verwendete serielle Schnittstelle feststellen. Ist sie nicht als einer der letzten Einträge aufgeführt, stecken wir den USB-IEC kurz ein und wieder aus und wiederholen den Aufruf des Befehls.

Für unseren beispielhaft verwendeten Iskra MT171 müssen wir die Schnittstelle wie folgt einstellen:

- Baudrate: 300
- Data Bits: 7
- Flow Control: None
- Parity: Even
- Open Mode: Read/Write
- Stop Bits: 1
- Device: /dev/ttyUSB0

Mit „Open" öffnen wir die Schnittstelle (Bild 2).

Bei unserem im Beispiel verwendeten Zähler müssen wir mit der Befehlsfolge *2F 3F 21 0D 0A* zunächst den Zähler ansprechen, der sich dann mit seiner Kennung (u. a. Hersteller, Typ und aktuelle maximale Baudrate) zurückmeldet (Bild 3).

Tipp:
Wir benutzen für unser Beispiel durchgehend hexadezimale Zahlen als Befehle. Die Folge

2F	3F	21	0D	0A
/	?	!	CR	LF

bedeutet übertragen in ASCII-Zeichen die unter den hexadezimalen Zahlenwerten aufgeführte ASCII-Folge, wobei CR für Carriage Return und LF für Line Feed stehen. Bei der reinen Übertragung in

Bild 1: Der USB-IEC an einem Stromzähler Iskra MT681 mit aktivierter optischer Schnittstelle

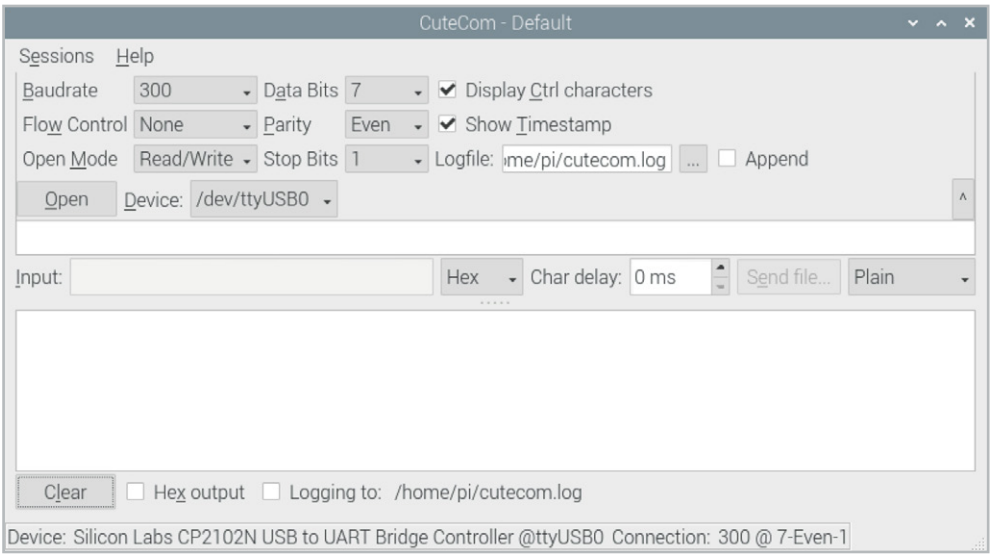


Bild 2: Konfiguration der Schnittstelle für den Iskra MT171

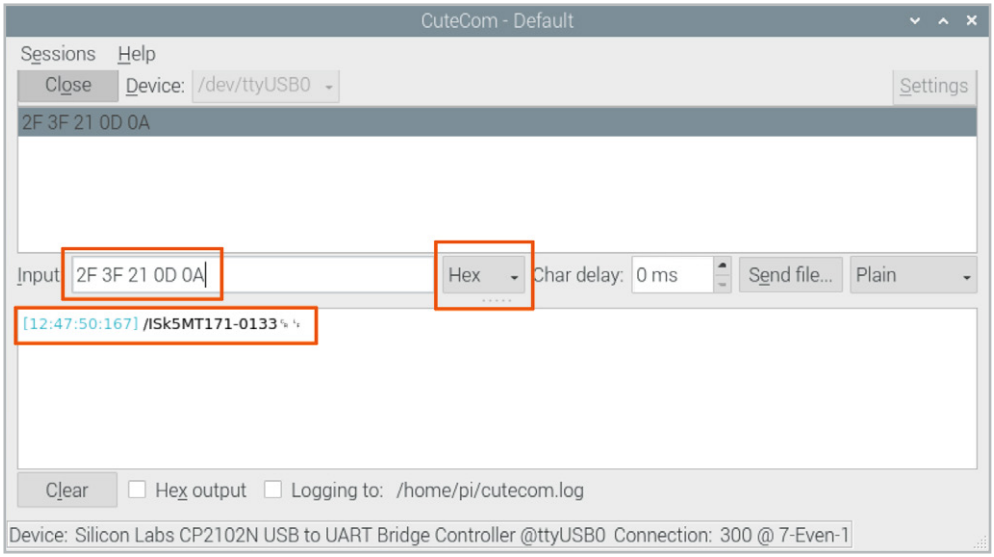


Bild 3: Rückmeldung des Stromzählers

hexadezimalen Zahlen gibt es vor allem hinsichtlich der abschließenden CR- und LF-Zeichen weniger Verwirrung, aber das mag Geschmackssache vor allem auch hinsichtlich des jeweils verwendeten Terminal-Programms sein.
Bei den Quellen im Internet ist nicht immer eindeutig, wie die Befehlsfolge genau im Terminal-Programm einzugeben ist, von daher sind hier unter Umständen mehrere Versuche notwendig.

Können wir mit unserem Stromzähler auf diese Art und Weise kommunizieren, ist die größte Hürde bereits genommen. Nun müssen wir noch den Befehl zur Ausgabe der Daten an den Zähler senden. In unserem Fall ist das recht zeitkritisch, da dieser Befehl kurz nach der Rückmeldung des Zählers erfolgen muss.

In unserem Terminalprogramm geben wir:

06 30 30 30 0D 0A

unmittelbar nach der Rückmeldung ein und erhalten in der Ausgabe des Terminalprogramms die verfügbaren Datensätze (Bild 4). Unter Umständen muss man dies mehrfach versuchen, bevor man eine Rückmeldung erhält.

Nutzt man lieber Windows, sind die Einstellungen bei HTerm in den Screenhsots in Bild 5 und Bild 6 zu sehen.

Diesen Ablauf der Kommunikation mit dem DO-Protokoll des Iskra MT171 werden wir später nutzen, wenn es darum geht, die Daten in Node-RED aus dem Stromzähler auszulesen.

Schritt 3 Streng nach Protokoll

Im obigen Beispiel des Stromzählers MT171 wird bei diesem das DO-Protokoll mit einer Aufwecksequenz verwendet, für das wir jetzt einen sogenannten Flow mit allen Bestandteilen in Node-RED erstellen werden. Im Gegensatz zum Stromzähler vom Typ MT681 (s. Schritt 4) des gleichen Herstellers, bei dem die optische Schnittstelle bereits aktiviert ist und die Daten im SML-Protokoll ausgegeben sind, gibt es hier kein bereits fertiges Node-RED-Modul zum Auslesen.

Doch bevor wir Node-RED starten, werfen wir einen Blick auf die zurückgegebenen Daten:

0-0:C.1.0*255(46704565)
1-0:0.0.0*255(784000-2900038)
1-0:0.2.0*255(V1.0)
1-0:1.8.0*255(094504 kWh)
1-0:2.8.0*255(055653 kWh)

Nach dem OBIS-Kennziffernschema (siehe auch [2]) sehen wir hier in den ersten drei Zeilen Angaben zum Zähler (Zählernummern, Version). Erst darunter

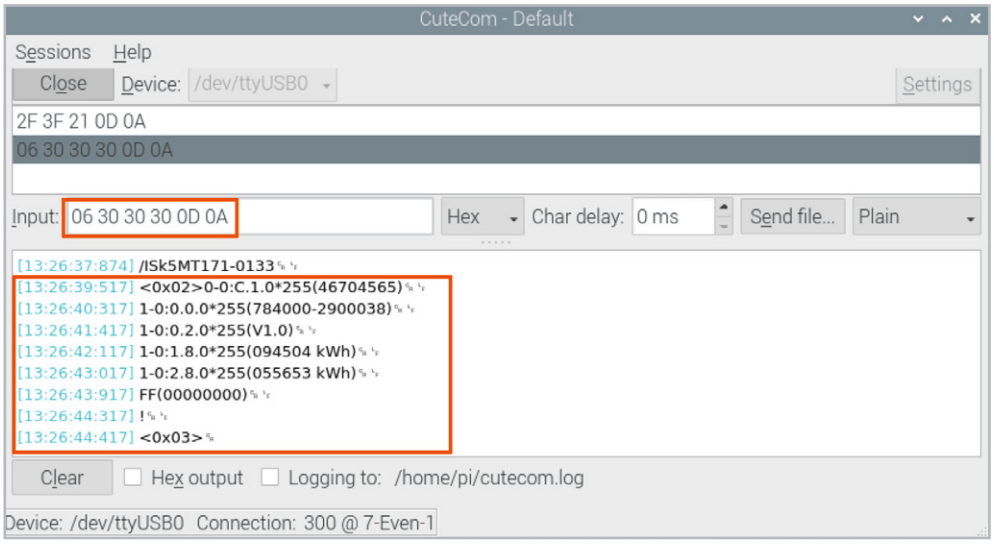


Bild 4: Verfügbare Datensätze des MT171

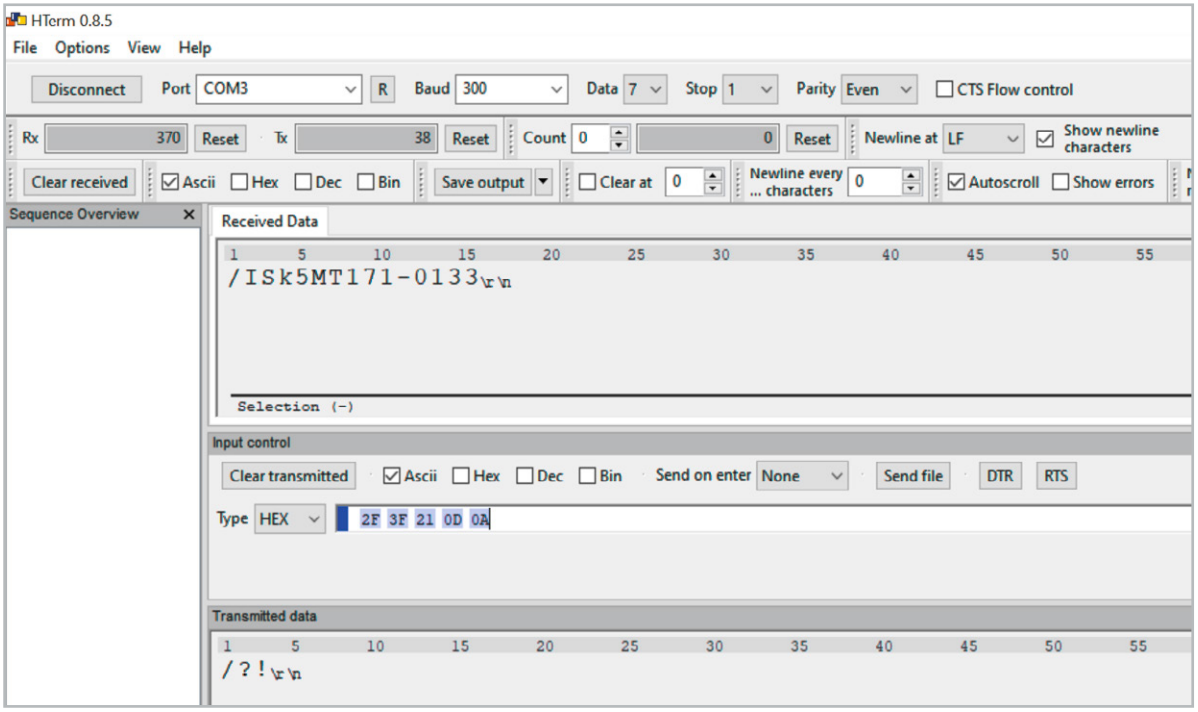


Bild 5: Initialer Befehl zur Kommunikation bei HTerm …

wird es mit echten Messwerten interessant. Unter 1.8.0 wird der Zählerstand „Bezug, gesamt" über alle Tarife angezeigt. Da es sich um einen Zweirichtungszähler handelt, folgt darunter unter 2.8.0 die Summe zur Stromeinspeisung in das Netz. Diese beiden Werte sind im Folgenden für uns von besonderem Interesse.

Datenanalyse

Mit diesem Wissensstand starten wir Node-RED wie gewohnt auf unserem Raspberry Pi und loggen uns unter der IP des Kleinrechners im Node-RED-Editor ein. Eine ausführliche Einführung zur Installation und den ersten Schritten in Node-RED gibt es im ELVjournal 4/2020 [3].

Für unsere Anwendung mit dem Zugriff auf die serielle Schnittstelle benötigen wir das Modul node-red-node-serialport, das wir mit der Palettenverwaltung installieren.

Im Flow erstellen wir zunächst einen Inject-Node, der den Aufweck- und Auslesevorgang anstoßen wird. Danach müssen wir ein bisschen tricksen: Da wir zwei hexadezimale Befehlsfolgen (s. o.) in einem zeitkritischen Abstand senden wollen, bedienen wir uns eines Trigger-Nodes, in den wir die beiden Folgen in Form eines Buffer-Arrays zeitversetzt senden.

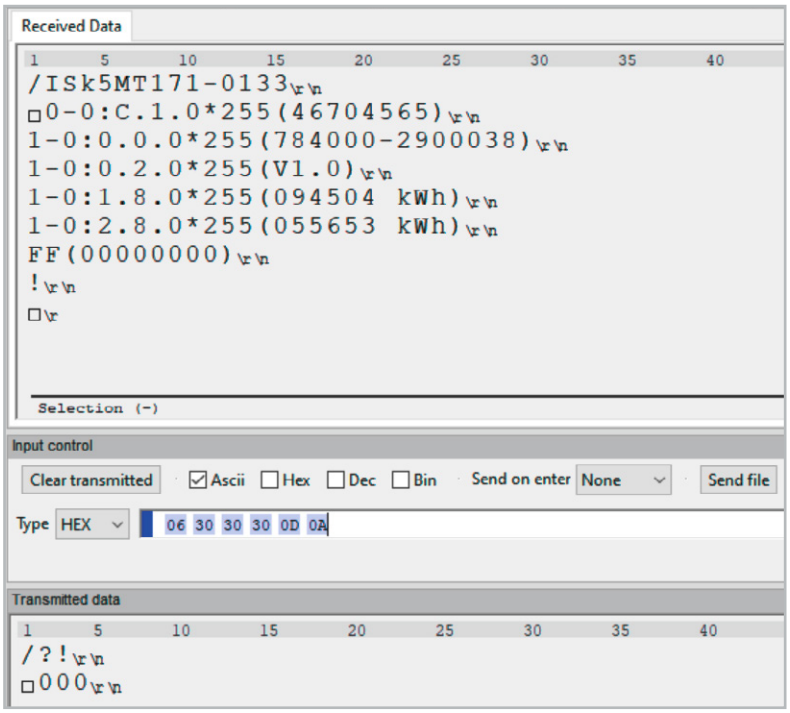


Bild 6: … und anschließende Ausgabe der Daten

Hier sind unter Umständen je nach verwendetem Stromzähler Anpassungen bzw. Feinschliff notwendig (Bild 7).

Natürlich ließe sich das noch perfektionieren – eigentlich müsste die erste Antwort vom Zähler abgewartet und ausgewertet werden. Mit dem Date-Readout-Befehl erfolgt dann auch die Umschaltung der Baudrate.

Auf den Trigger-Node folgt die Konfiguration des Serial-Request-Nodes nach den Daten aus dem oben gezeigten Ablauf der seriellen Kommunikation mit dem Stromzähler (Bild 8).

Hängt man an diesen Flow einen Debug-Node an, erhält man bereits die Ausgabe wie im Terminal. Wie immer kann man in Node-RED die Daten jetzt auswerten, anzeigen und/oder weiterleiten. Ein Beispiel-Flow (Bild 9) liegt bei diesem Beitrag [8] zum Download bereit.

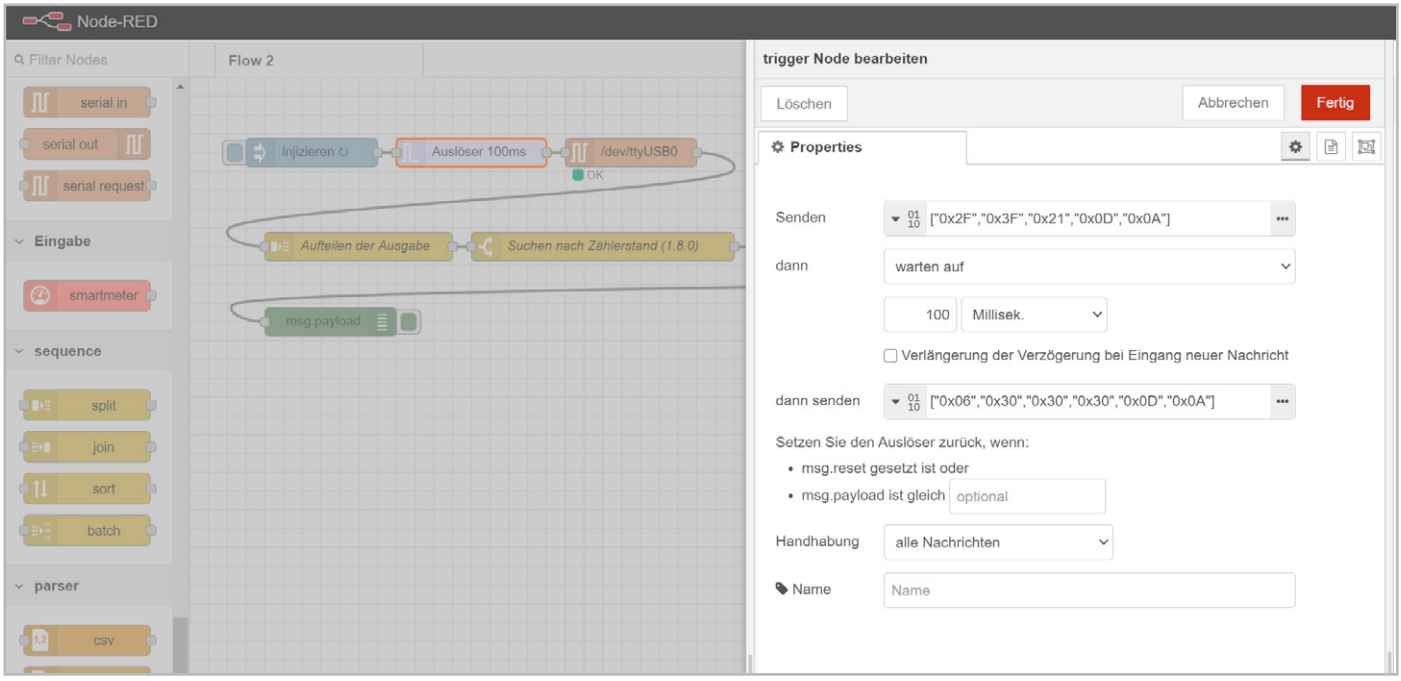


Bild 7: Konfiguration des Trigger-Nodes

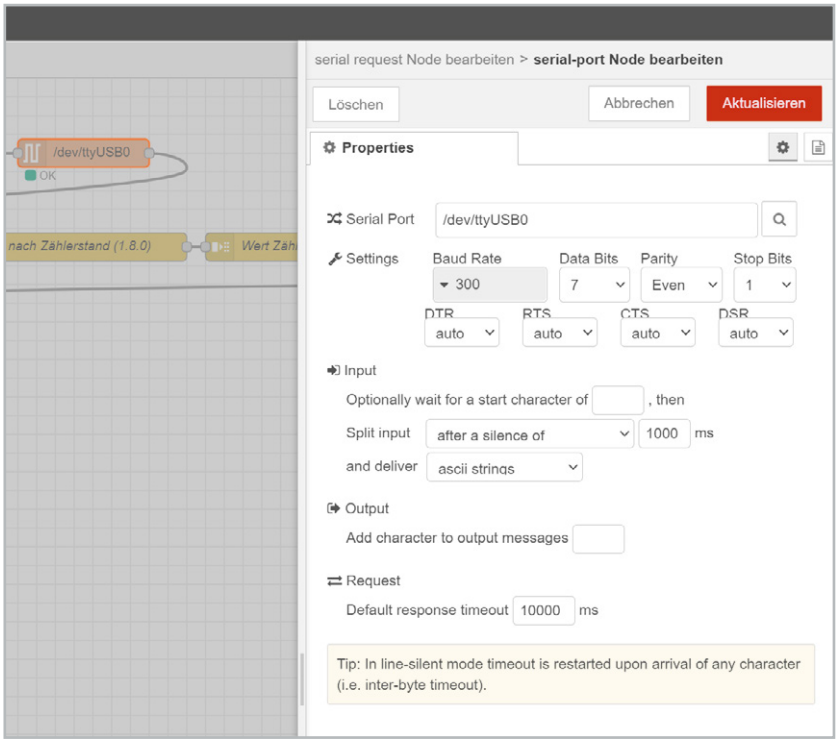


Bild 8: Konfiguration des Serial-Request-Nodes

Schritt 4 SML mit Smartmeter-Node

Haben wir einen Stromzähler mit ständig aktivierter Ausgabe, der mit dem DO- oder SML-Protokoll arbeitet, können wir das eigens für diesen Zweck entwickelte Node-RED-Modul node-red-contrib-smartmeter nutzen. Wir installieren dieses und ziehen den Smartmeter-Knoten in den Editor. Diesen müssen wir zunächst konfigurieren – auch hier bleibt es uns nicht erspart, die Eigenschaften der seriellen Schnittstelle herauszufinden und das verwendete Protokoll einzustellen (Bild 10).

Hier reicht ein Debug-Node und die Zählerdaten werden nach dem OBIS-Kennziffernschema ausgegeben. Bei dem verwendeten Zähler Iskra MT681 wird nicht nur der Zählerstand, sondern auch die momentane Leistung angezeigt. So lassen sich neben der automatisierten Ablesung des Zählers beispielsweise auch Analysen über den Verbrauch im (Tages-)Verlauf anstellen.

Im Download-Bereich dieses Beitrags [8] finden Sie einen Beispiel-Flow mit der Extraktion der Messwerte, Weiterleitung per MQTT und einer schicken Anzeige im Dashboard, die sich lokal im Browser oder von außen per VPN auch auf dem Smartphone abrufen lässt (Bild 11).

Fazit

Mit dem Bausatz USB-IEC-Interface und dem Software-Tool Node-RED lassen sich in wenigen Schritten Stromzähler, aber auch andere Geräte mit optischer Schnittstelle auslesen. Neben der Auswertung der Messdaten kann man diese auch

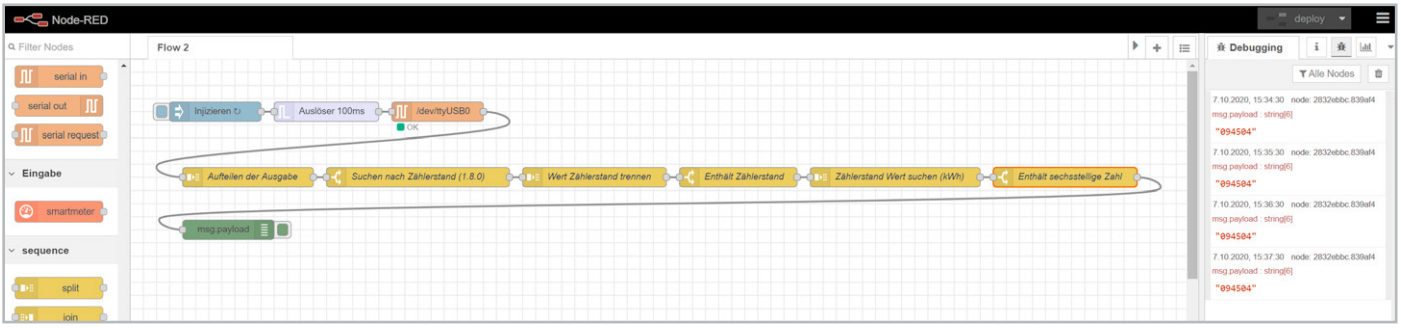


Bild 9: Beispiel-Flow, der den Messwert zum Strombezug aus dem Datensatz extrahiert

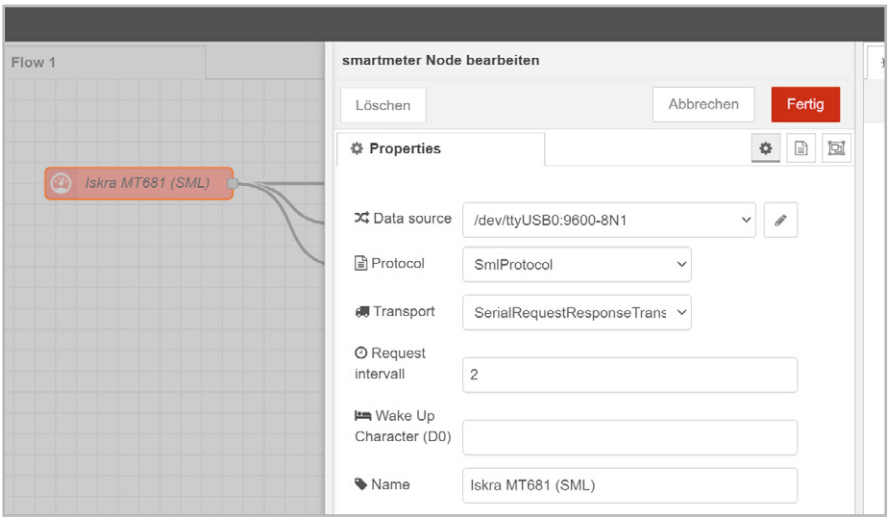


Bild 10: Konfiguration des Smartmeter-Nodes

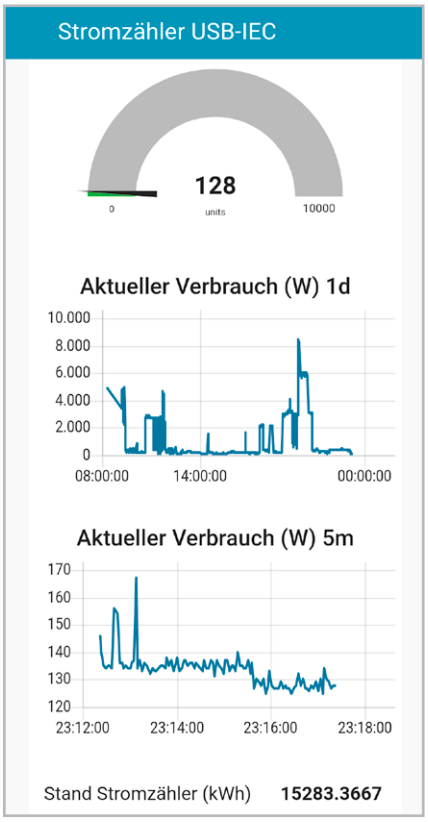


Bild 11: Anzeige der Zählerdaten per Node-RED-Dashboard auf dem Smartphone

ansprechend darstellen und zur weiteren Verarbeitung weiterleiten. Der nächste Schritt zur automatisierten Analyse von bestimmten Verbrauchern anhand ihres Stromverbrauchs ist da nur noch einen Schritt (mit etwas künstlicher Intelligenz) entfernt.

Können Sie erkennen, welche Maschine sich hinter dem Muster in der 5-Minuten-Ansicht zum „Aktuellen Verbrauch (W) 5 m" im Aufmacherfoto des Beitrags versteckt?

Folgen Sie uns auf unseren Social-Media-Kanälen auf Facebook (facebook.com/elvelektronik) und Twitter (twitter.com/elvelektronik).

Nach Erscheinen des ELVjournals verraten wir dort die Auflösung.

Weitere Infos:

[1] ELVjournal 2/2016: Energieverbrauch im Blick – Energie-Sensor für Smart Meter
www.elv.com: Artikel-Nr. 205621
[2] ELVjournal 4/2019: Fachbeitrag: Rente für Ferraris-Zähler – Digitale Stromzähler mit SML-Protokoll auslesen
www.elv.com: Artikel-Nr. 250925
[3] ELVjournal 4/2020: Programmieren (fast) ohne Code – Node-RED als universelles Prototyping-Tool, Teil 1
www.elv.com: Artikel-Nr. 251410
[4] ELVjournal 5/2020: Programmieren (fast) ohne Code – Node-RED als universelles Prototyping-Tool, Teil 2
www.elv.com: Artikel-Nr. 251516
[5] Raspberry Pi 4 Model B, 2GB RAM: www.elv.com: Artikel-Nr. 250569
[6] https://files2.elv.com/public/14/1421/142148/Internet/142148_zaehlerliste_data.pdf
[7] HTerm Download: http://der-hammer.info/pages/terminal.html
[8] Download der Flows zum USB-IEC: www.elv.com: Artikel-Nr. 251605

Alle Links finden Sie auch online unter: de.elv.com/elvjournal-links